Gaussian Processes and CNNs: The utility of Prediction Variance

Joshua Send [1]

Abstract

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer pretium lectus eget lacus fermentum, ac tempus nibh convallis. Aliquam in est dolor. Phasellus lobortis augue a mattis cursus. Vestibulum malesuada blandit enim, at convallis sem imperdiet eu. Sed nec elementum turpis.

1. Introduction

Convolutional Neural Nets (CNNs) have seen a steep rise in use for a wide range of computer vision and machine learning tasks since Krizhevsky's success at the ImageNet challenge in 2012 (Krizhevsky et al., 2012). Given enough training data, they excel in domain specific applications and are efficient to train, but may confidently mis-predict classes, provide no feedback to users about variability of predictions, and have been shown to be vulnerable to adversarial attacks (Szegedy et al., 2013). On the other hand, Gaussian Processes are capable of classification and providing a variance about the prediction. This can be interpreted as a confidence in the probability of a prediction.

This work combines CNNs with Gaussian Processes (GPs), using the second to last layer of the network as the input to the GP. This combination takes advantage of neural networks' ability to approximate high dimensional data efficiently; GPs can then produce Bayesian uncertainty estimates along with predictions.

The goal of this work is to explore how useful the variance produced by a GP is, in terms of interpretability and enhancing prediction accuracy. Focus will be on image classification using the MNIST (LeCun et al., 1998) and N-MNIST (Basu et al., 2017) datasets, along with adversarially perturbed MNIST images.

[1]University of Cambridge. Correspondence to: Joshua Send <js2173@cam.ac.uk>.

2. Background

2.1. Convolutional Neural Nets

CNNs are composed a layers of neurons[1] with associated weights and activation functions. A CNN takes raw input (images) and reduce them to higher level feature representations. Training is done via backpropagation, minimizing some loss function by adjusting the weights in the network. This is an $O(n)$ operation repeated some number of iterations.

To use CNNs for classification, the second to last layer is fully connected to a layer of m units, where m is the number of classes. The following *softmax* activation is then applied:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{k=m} e^{z_j}}$$

Combined with categorical-cross entropy loss during training, the resulting values between 0 and 1 from each of the m units represent the posterior distribution over the classes (Bridle, 1990). In other words, we obtain the probability of any class using a fully connected softmax layer. Note that this method reinforces large values and squashes small ones due to the exponentiation.

2.2. Gaussian Processes

Gaussian processes are non-parametrized models of functions, defined by a mean and covariance function. The latter determines how related two points in space are – generally the further away, the less related. The covariance function is built out of a set of *kernels*, and generates a covariance matrix. By incorporating the mean and covariance functions, along with training data, GPs predict both a mean and a variance for input points.

[TODO image showing uncertainty increasing away from training data]

One of GPs' key benefits is that little hyperparameter tuning is needed – since the model is fully Bayesian, parameters can be learned directly by maximizing marginal likelihood. However, traditional GPs are costly to train – covariance matrices are $O(n^2)$ in size, and inversion and decomposition

[1]A very coarse approximation to biological neurons

operations have an $O(n^3)$ cost, where n is the number of training samples.

To deal with the issue of scalability[2], various approximation methods have been developed. One approach that is used here is called Sparse Variational Gaussian Processes (Hensman et al., 2015). This classification approach effectively uses a set of *inducing points* optimized from the training data as an estimate of the full data set; these are then used to fit a GP.

Gaussian processes classification is described in Rasmussen (2004). This book also describes a variety of kernels, the choice of which strongly affects performance – some are explored in Section 5.

2.3. Adversarial Attacks

In this paper, an adversarially perturbed MNIST dataset is tested. The finite-gradient sign method (Goodfellow et al., 2014) (FGSM) is used to perform non-targeted attacks. This approach requires access to the trained model to maximize a loss function, performing gradient ascent away from correct classifications by adding small amounts of noise at each step. This noise is controlled by a parameters ϵ.

Other attacks, such as Jacobian-based saliency maps (Papernot et al., 2016b) are not explored here, but the work could easily extended to include it.

3. Related Work

The key component of this work is using a fully Bayesian inference method to derive interpretable confidences, as well as variances for predictions. Some work exists along these lines, adapting CNNs to produce variances as well. One notable contribution is dropout-based approach (Gal & Ghahramani, 2016), using dropout at *test* time to sample a variety of predictions and computing a mean and variance from this.

Another approach is using fully Bayesian Neural Networks, an idea established in the 90s (MacKay David, 1992). These place a distribution over each parameter in the network, making inference and calculating the full posterior distribution essentially intractable. More recently there has been work towards this end (Blundell et al., 2015) that is back-propagation and GPU compatible. Other work has applied similar ideas to recurrent neural networks as well (Zhu & Laptev, 2017).

The most directly related work from Bradshaw et. al. (Bradshaw et al., 2017) who investigate GPDNNs (Gaussian Process Deep Neural Nets) in a similar manner to this paper. However, they emphasize end-to-end trained models, comparing neural nets with softmax classification to ones with a GP replacing the softmax layer and retraining from scratch. In contrast, this work explores using pre-trained neural nets in combination with gaussian processes which are trained post-hoc.

This work will evaluate the normal CNN and the GP model on adversarially perturbed MNIST images, showing modest gains and better interpretability in the GP. Most methods for protecting machine learning models from such attacks are either based on data augumentation (which does not scale), or training multiple networks with high overhead. Recently alternative approaches have been proposed, such as using GANs (Samangouei et al., 2018)(Zantedeschi et al., 2017).

4. Implementation

The core libraries used were Keras (Chollet et al., 2015) with a TensorFlow[3] backend for the MNIST CNN. GPFlow (Matthews et al., 2017) was used to implement the gaussian process model due to its support for large, high dimensional training data[4]. Scipy's machine learning toolkit was also tested but was found to be unsuitable.

The models initially tested are

1. Keras MNIST CNN trained on 60,000 training images.
2. 3 different GPs trained on the same images after feature extraction through the CNN's first 6 layers (ie. excluding the fully connected softmax layer). The GPs tested will be explained in 5.1.

TODO MNIST CNN architecture figure

* Approximate trainining time for GP (=¿ discussion?) * Predict time for GP (=¿ discussion?)

* Train, test sizes across MNIST, NMNIST, Adverserial (Papernot et al., 2016a), and image sizes (28x28, grayscale) * Balanced datasets? (! TODO)

5. MNIST

This section focuses on the MNIST test data set, consisting of 10,000 examples, each 28x28 pixels.

5.1. Accuracy of Various Models

Figure TODO shows the test-set accuracy of the CNN, along with three different Gaussian Processes: one using a *Polynomial* kernel, one using the *Matern12* kernel, and the last a

[2]The training data here is a 60000x128 vector which is formed into a square matrix

[3]https://www.tensorflow.org/

[4]GPFlow implements sparse variational GPs described in Section 2.2 as well as batched training

*Matern32*Linear* kernel (referred to as 'combined' kernel from now on). For explanation of these kernels please refer to **?**.

TODO NSERT FIGURE BAR with CNN, Polynomial, Matern12, combined kernels

These kernels were chosen out of 17 different single and combined kernels. It will be shown later that the *Polynomial* kernel performs well in adversarial situations, as does *Matern12*, while the combined kernel appears to inherit properties from its two components: *Linear* kernels exhibit strong performance on non-adverserial datasets, while *Matern32* performs worse across the board but has an interesting property which will be explored in Section 5.5.

While exploring the GP design space, many configurations were tested. The most influential by far was the number of inducing points used. All kernels in this paper were created using every 25th data point as a possible inducing point and allowed to be automatically optimized during training. Minibatch sizes were set at 8000. A gaussian noise kernel was added to all kernels with constant variance 0.1, though its inclusion nor its variance have much impact on results.

5.2. Distribution of Variance

The previous section simply compares the performance of *softmax* versus a GP as a feature classifier. The goal of this paper is to examine the utility of prediction variance, which the CNN does not provide. The following figures examine the distribution of variance for correctly, and incorrectly classified examples using the combined kernel.

FIGURES TODO

These plots clearly show that the variance of the predicted class is higher than for those that are predicted correctly. This aligns with our expectations that correctly classified examples are generally confidently predicted, while inputs that are incorrectly classified have a lower confidence (higher variance). Thus, as a use case, a human operator seeing high variance should be suspicious of an incorrect classification [5].

5.3. SVGP Variance

The GP implementation used in this paper is the sparse variational gaussian process (SVGP) implemented in GPFlow. It is worth double checking that the variance being predicted makes sense and is providing additional information.

TODO FIGURE

The above figure shows that, actually, with this technique and implementation, the confidence predicted completely determines the predicted variance. In fact, a peek into the implementation reveals that variance σ is calculated from the predicted mean p as $\sigma = p - p^2$.

Note that this observation does not invalidate the usefulness of variance – it simply says that there is no point getting both mean and variance from the SVGP. Calculating the variance and in turn standard deviation is still a useful indicator for explaining results, and later it shall be used to merge *softmax* and GP predictions.

5.4. Examining Misclassifications

While noting that all GP models and the CNN achieve very high classification accuracies, it is worth further examining the misclassifications of each type of model. I again focus on results from the combined kernel as representative of all three GP models.

First note that the CNN is slightly more confident when mis-predicting than the GP. Since we established that variance is correlated directly with prediction confidence, this is what we expect, though the CNN results have no such interpretation.

Table 1. Mean and Standard Deviation of probability of incorrect predicted classes for the combined kernel GP and the CNN.

STATISTIC	GP	CNN
MEAN	74.5%	75.3%
STANDARD DEVIATION	17.7%	18.5%

Next, we can inspect the particulars of mis-classifications.

TODO figure VENN DIAGRAM

The Venn diagram in Figure TODO illustrates that both models largely fail on the same test data. These can be seen as the truly difficult or ambiguous images, and nothing much can be done about them.

TODO figure both misclassified, CNN probs, GP probs

However, more interesting is the non-overlapping region. Though here there are few examples out of the entire dataset of 10000, in other settings these disagreeing classification regions are larger.

TODO figure of CNN incorrect MNIST, CNN probabilities, GP probs. TODO figure of GP incorrect MNIST, CNN probabilities, GP probs.

From Figures TODO and the corresponding probability distributions we see there might be hope for boosting accuracy by combining the results from the CNN and the GP. This inspires the hybridized model in the next section.

[5] This is particularly useful in medical and related fields where a mistake is extremely costly.

5.5. Hybridization

Because we have a sort of ensemble of classifiers when using both the CNN and a GP, we can use the extra bit of information during disagreements to try to choose the best result, steered by the variance predicted by the GP.

We can devise a heuristic for merging results as follows. The driving principle is that we trust the Gaussian Process model more than the CNN, since the CNN gives no indicator if how reliable its result is.

If the GP predicts a higher probability than the CNN's best prediction, we default to the GP's result. If the top CNN prediction is within d standard deviations from the same GP prediction, we accept the CNN prediction (in practice this ensures the GP agrees the probability is plausible given its prediction variance). A stronger acceptance criterion that can be used is that we require both the top CNN prediction agree with the GPs corresponding probability, as well as the opposite: the top GP's prediction is within d standard deviations from the CNN's prediction for the same class.

There are several points to discuss regarding this merge: when accepting a CNN prediction, we lose a meaningful variance and the resulting interpretability. However this could easily be fixed by flagging the result as uncertain due to disagreement. Secondly, some kernels lend themselves to this blending better than others. Through experimentation the *Matern32*, and in turn the combined *Linear*Matern32*, kernels were found to provide the best results ***WHY***. Lastly, other merging criteria are possible and could be explored.

From here on I also present Hybridized accuracy results using the combined kernel with an acceptance criteria of $d = 0.5$ standard deviations.

* Describe both criteria, pros and cons of each

6. N-MNIST

6.1. White Noise + MNIST

(Sample image)

* Accuracy across, CNN, Matern12, Poly, Linear*Matern32, Hybridized * Distribution of correct, incorrect classification Variances for Linear*Matern32

6.2. Blurred MNIST

(Sample image) * Accuracy across, CNN, Matern12, Poly, Linear*Matern32, Hybridized * Distribution of correct, incorrect classification Variances for Linear*Matern32

6.3. White Noise and Low Contrast MNIST

(Sample image) * Accuracy across, CNN, Matern12, Poly, Linear*Matern32, Hybridized * Distribution of correct, incorrect classification Variances for Linear*Matern32

7. Adverserial Attacks

* Brief description of FSGM, that it uses the trained CNN to generate adverserial examples with some epsilon

* Accuracy across models as epsilon varies

* Distribution of correct, incorrect classification Variances for Linear*Matern32 for eps=0.2

8. Conclusion

* Usefulness of variance * Resistance to adverserial attacks * kernel types * reference automatic kernel building? * hybridization effectiveness/ineffectiveness * =¿ ensembles of GP's and kernels? =¿ more interesting hybridization? * mention SVMs, outperform other methods when data is clean, read up on probabilistic outputs of SVMs?

Citations within the text should include the authors' last names and year. If the authors' names are included in the sentence, place only the year in parentheses, for example when referencing Arthur Samuel's pioneering work (**?**). Otherwise place the entire reference in parentheses with the authors and year separated by a comma (**?**). List multiple references separated by semicolons (**???**). Use the 'et al.' construct only for citations with three or more authors or after listing all authors to a publication in an earlier reference (**?**).

Authors should cite their own work in the third person in the initial version of their paper submitted for blind review. Please refer to Section **??** for detailed instructions on how to cite your own papers.

Use an unnumbered first-level section heading for the references, and use a hanging indent style, with the first line of the reference flush against the left margin and subsequent lines indented by 10 points. The references at the end of this document give examples for journal articles (**?**), conference publications (**?**), book chapters (**?**), books (**?**), edited volumes (**?**), technical reports (**?**), and dissertations (**?**).

Alphabetize references by the surnames of the first authors, with single author entries preceding multiple author entries. Order references for the same authors by year of publication, with the earliest first. Make sure that each reference includes all relevant information (e.g., page numbers).

8.1. Software and Data

Code, results, and this report can be found at:

https://github.com/flyingsilverfin/CNN_GP_MNIST

Please note that intermediate results are not saved but can be recomputed, and that some of the configurations are specific to the development machine.

Acknowledgements

Do not include acknowledgements in the initial version of the paper submitted for blind review.

If a paper is accepted, the final camera-ready version can (and probably should) include acknowledgements. In this case, please place such acknowledgements in an unnumbered section at the end of the paper. Typically, this will include thanks to reviewers who gave useful comments, to colleagues who contributed to the ideas, and to funding agencies and corporate sponsors that provided financial support.

References

Basu, Saikat, Karki, Manohar, Ganguly, Sangram, DiBiano, Robert, Mukhopadhyay, Supratik, Gayaka, Shreekant, Kannan, Rajgopal, and Nemani, Ramakrishna. Learning sparse feature representations using probabilistic quadtrees and deep belief nets. *Neural Processing Letters*, 45(3):855–867, 2017.

Blundell, Charles, Cornebise, Julien, Kavukcuoglu, Koray, and Wierstra, Daan. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

Bradshaw, John, Matthews, Alexander G. de G., and Ghahramani, Zoubin. Adversarial Examples, Uncertainty, and Transfer Testing Robustness in Gaussian Process Hybrid Deep Networks. pp. 1–33, 2017. URL http://arxiv.org/abs/1707.02476.

Bridle, John S. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neurocomputing*, pp. 227–236. Springer, 1990.

Chollet, François et al. Keras. https://github.com/keras-team/keras, 2015.

Gal, Yarin and Ghahramani, Zoubin. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pp. 1050–1059, 2016.

Goodfellow, Ian J, Shlens, Jonathon, and Szegedy, Christian. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Hensman, James, Matthews, Alexander G de G, and Ghahramani, Zoubin. Scalable variational gaussian process classification. 2015.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

LeCun, Yann, Bottou, Léon, Bengio, Yoshua, and Haffner, Patrick. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

MacKay David, JC. A practical bayesian framework for backprop networks. *Neural computation*, 1992.

Matthews, Alexander G. de G., van der Wilk, Mark, Nickson, Tom, Fujii, Keisuke., Boukouvalas, Alexis, León-Villagrá, Pablo, Ghahramani, Zoubin, and Hensman, James. GPflow: A Gaussian process library using TensorFlow. *Journal of Machine Learning Research*, 18(40): 1–6, apr 2017. URL http://jmlr.org/papers/v18/16-537.html.

Papernot, Nicolas, Goodfellow, Ian, Sheatsley, Ryan, Feinman, Reuben, and McDaniel, Patrick. cleverhans v1.0.0: an adversarial machine learning library. *arXiv preprint arXiv:1610.00768*, 2016a.

Papernot, Nicolas, McDaniel, Patrick, Jha, Somesh, Fredrikson, Matt, Celik, Z Berkay, and Swami, Ananthram. The limitations of deep learning in adversarial settings. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pp. 372–387. IEEE, 2016b.

Rasmussen, Carl Edward. Gaussian processes in machine learning. In *Advanced lectures on machine learning*, pp. 63–71. Springer, 2004.

Samangouei, Pouya, Kabkab, Maya, and Chellappa, Rama. Defense-gan: Protecting classifiers against adversarial attacks using generative models. 2018.

Szegedy, Christian, Zaremba, Wojciech, Sutskever, Ilya, Bruna, Joan, Erhan, Dumitru, Goodfellow, Ian, and Fergus, Rob. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Zantedeschi, Valentina, Nicolae, Maria-Irina, and Rawat, Ambrish. Efficient defenses against adversarial attacks. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 39–49. ACM, 2017.

Zhu, Lingxue and Laptev, Nikolay. Deep and confident prediction for time series at uber. In *Data Mining Workshops (ICDMW), 2017 IEEE International Conference on*, pp. 103–110. IEEE, 2017.

A. Do *not* have an appendix here

Do not put content after the references. Put anything that you might normally include after the references in a separate supplementary file.

We recommend that you build supplementary material in a separate document. If you must create one PDF and cut it up, please be careful to use a tool that doesn't alter the margins, and that doesn't aggressively rewrite the PDF file. pdftk usually works fine.

Please do not use Apple's preview to cut off supplementary material. In previous years it has altered margins, and created headaches at the camera-ready stage.